Docket: NH-1

PATENT APPLICATION

Attribute Tree for Modeling Value of Multi-Attribute Products/Services for an Online Trading System

INVENTORS:

Ravi V. Condamoor
1382 FisherHawk Drive
Sunnyvale, CA 94087
A citizen of USA

Ankur Datta Sharma
1065 Marigold Court
Sunnyvale, CA 94086
A citizen of USA

Neelakantan Sundaresan
492 Capitol Village Circle
San Jose, CA 95136
A citizen of India

ASSIGNEE:

NehaNet Corp.
2355 Paragon Drive, Suite E
San Jose, CA 95131

Stuart T. Auvinen
429 Twenty-Sixth Ave.
Santa Cruz, CA 95062-5319
(831) 476-5506
Reg. No. 36,435
PTO Customer # 23933

23933

PATENT TRADEMARK OFFICE

PATENT

NH-1

05/23/00 3:14 PM

ATTRIBUTE TREE FOR MODELING VALUE OF MULTI-ATTRIBUTE PRODUCTS/SERVICES FOR AN ONLINE TRADING SYSTEM

RELATED APPLICATION

A provisional patent application entitled "System and Method of Modeling Value in an Electronic Trading System" was filed on 4/17/00 by the same inventors for the present application, U.S. Provisional Appl. No. 60/198,125.

FIELD OF THE INVENTION

This invention relates to electronic commerce systems, and more particularly to value and pricing of complex products.

BACKGROUND OF THE INVENTION

The widespread adoption of the Internet has facilitated new relationships among buyers and sellers. Internet auctions and name-your-price services have empowered the consumer, offering deeper discounts with minimal consumer effort. More recently, business buyers have taken note of the price discounting on the Internet and have been seeking to cut costs through Internet-based purchasing. Business-to-Business (B2B) exchanges have sprung up in many markets to encourage competition among suppliers. Transaction costs are dramatically lowered since paper-based quotes and purchase orders are replaced by electronic documents.

While such electronic commerce exchanges are useful, often the products sold are complex. Many variations or configurations of a product can be sold. In the past, fixed prices were usually assigned to each configuration of a product or each product option, and a buyer merely had to add up the cost of the various options desired. However, these electronic exchanges often do not sell products based on a fixed price schedule. Complex pricing models may be based on volume purchased by a single buyer, or aggregation of demand by many buyers. The price may even be determined by auction, or by some other negotiation between the buyer and seller. The negotiations may even be carried out by computer programs that follow pre-programmed rules.

When complex pricing is combined with many possible configurations of a product, the complexity of any pricing models increases dramatically. When many buyers aggregate demand, each of the buyers may prefer a different configuration of the base product, resulting in a confusing demand mix. Modeling such mixed demand is problematic.

The various product configurations can be described using multiple product attributes. Multi-attribute models are known in relational database and Online Analytical Processing (OLAP) systems. Multi-attribute and multi-attribute electronic commerce systems include multi-attribute auctions, negotiations, and contracts in electronic procurement and other areas.

In "Multi-attribute auctions for electronic procurement", by Bichler, Kaukal, & Segev (IAC's Workshop on Internet Based Negotiation Technologies, March 18-19, 1999 Yorktown Heights, NY), the authors describe a system wherein price is one of the many attributes in determining the value of a product. Other attributes may optionally include shipping-handling costs, warranty etc. Here the value of a product is computed in terms of a utility function which is the weighted sum of the utility units corresponding to each attribute.

While such multi-attribute models are useful, a multi-attribute model more suited for electronic exchanges is desired. A way to express the many options or configurations of products in a product family is desired. It is desired to receive, store, and compare multiple product configurations from many buyers and sellers. A simple yet powerful structure to store many attributes of a product family is desired.

SUMMARY OF THE INVENTION

The present invention describes a system, method, and apparatus for participants in an electronic commerce application to model Value as a Complex Property. Value is modeled using the notion of Attributes.

The system includes mechanisms for

1. Naming Attributes
2. Defining Attribute metadata
3. Giving Attributes Priorities
4. Specifying and working with Dependencies Between Attributes
5. Specifying and working with attribute ranges

Further, the system includes facilities for

1. Representing Attributes
2. Algorithms for Creating, Updating, and Looking up Attributes

Attributes are represented using a dependency tree structure. Each attribute has a set of possible values. Each node in the tree represents an attribute and one of its possible values. The parent of such a node is a node that represents an attribute that this one is dependent on. For an attribute A with possible values vA1, vA2,.. vAk and an attribute B that is dependent on it with possible values vB1, vB2, ... vBl, each (A, vAi) node has l child (B, vBj) j = 1...l.

The attribute tree for any trading partner can grow to be exponential. For instance, for a trading partner with N attributes, A1, A2, ... AN, and M values for each of the attributes, where some of these attributes are dependent on others, in the worst case is a trading partner tree with (M**N - 1)/(M-1). A system is provided that can, using heuristics, reduce this size to a manageable size.

In general, a system is provided that allows a trading partner to breakdown and model a commodity in terms of a number of attributes and attribute values. In addition, the modeling system provides a facility to build an augmented products by adding new attributes or deleting existing attributes dynamically during the trade representing market conditions.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 shows a trading partner (TP) attribute Tree.

Figure 2 shows a path from the root to a leaf of an attribute tree.

Figure 3 shows how the data from multiple data sources or applications can be obtained implicitly.

Figure 4 is a block diagram of an attribute-tree builder.

Figure 5 shows one embodiment of a heuristic to create an attribute tree.

Figure 6 is a procedure for optimizing an attribute tree.

Figure 7 is a flowchart for a Business-to-Consumer (B2C) application of the attribute trees.

Figure 8 is a flowchart of an auction system enhanced by the attribute-structuring facility.

Figure 9 is a flowchart of an optimal product/seller selector that helps a bidder.

Figure 10 illustrates dynamic augmentation of a product with the attributes most desired by the potential buyers.

TERMINOLOGY

1. Electronic Commerce – Commerce conducted over an electronic medium
2. Electronic Exchange – A platform in which providers of goods or services buy, sell, and exchange those entities.
3. Utility Function – defines a function used to define the value of a product. Typically this is price in some units like dollars or pounds that the buyer is willing to buy at or the seller is willing to sell at.
4. Delta or Delta Value - A quantifiable vested interest of a trading partner. Different trading partners usually have different aspects of a trade that they value the most. Example: One might have price, the other may be concerned about quality, and the third might care about quantity and delivery time. The Delta is a manifestation of their preferences and can be positive or negative; positive indicates that the trading partner cares more about it and negative connotes the trading partner is less interested in that particular attribute of a trade.
5. Product (Product and Service used interchangeably)
6. Base Product Value or Base Value - is the utility function value (typically price) at which a trading partner starts valuing a product.
7. Delta Product Value - is the utility function value (typically price) of the product with the deltas for the attributes accounted for
8. Trade – A trade is an activity in which one or more parties provide services or goods to one or more parties at a pre-calculated or at a dynamically decided or negotiated cost.
9. Service category – Types of goods and services
10. Trading Partner – A buyer / seller / facilitator in a multi-party trade. Also a service provider.
11. Net Economic Value – Is the difference between the utility function value (typically price) at which a trading partner values a product and the utility function value (price) he actually trades at.

12. Attribute - An abstract data type that provides a mechanism for Trading Partners to describe their product/service to a finer level of granularity taking into account the strengths and weaknesses of the trading partners. Domain specific attributes will be defined by subclassing the base Attribute class. Ex: Date, Location, Weight are all attributes in the system
13. Attribute Instance - An instance of a valid Attribute. Ex: Instance of a Date attribute.
14. Attribute Value - Quantifiable values assigned to Attribute instances. Ex:)6/01/00 assigned to a Date attribute
15. Independent Attribute – Attribute with values whose delta is independent of any other attribute value
16. Dependent Attribute – Attribute with values whose delta is dependent on another attribute value
17. Attribute Range – Range over which an attribute's values are defined
18. Sets of Ranges – is a collection of disjoint ranges
19. Attribute Interval – defines a segment between two contiguous attribute values in a range
20. Attribute Metadata – Properties of the attribute class (ranges, casting function, creator, etc.) constitute the attribute metadata.
21. Decision Master – The process or object that helps a trading partner decide on participation in one or more trades optionally based upon past trade performances and other business intelligence information.
22. Attribute Value Tree – a dependency tree of attributes and their values for a product where any path from the root to a leaf represents a combination of attribute values chosen for the product.
23. Attribute Value Delta Tree – an Attribute Value Tree which in addition to the attributes and values keeps DELTA values. Delta values may be kept at each node or cumulatively for the paths at each leaf node.
24. TP Tree - A data structure containing the set of <attribute instance, attribute value> pairs and their respective Deltas of a Trading Partner. Though similar in

structure to a Trade Tree, a Trading Partner tree is typically a much larger set as it represents all services and their attributes that affect the final value of the service.

25. Trade Tree - A data structure containing the set of <attribute instance, attribute value> pairs and their respective Deltas for a Trade.

DETAILED DESCRIPTION

The present invention relates to an improvement in e-commerce. The following description is presented to enable one of ordinary skill in the art to make and use the invention as provided in the context of a particular application and its requirements. Various modifications to the preferred embodiment will be apparent to those with skill in the art, and the general principles defined herein may be applied to other embodiments. Therefore, the present invention is not intended to be limited to the particular embodiments shown and described, but is to be accorded the widest scope consistent with the principles and novel features herein disclosed.

The Decision Master (DM) is primarily responsible for making a decision on behalf of a Trading Partner (TP) in an Electronic Commerce System. The TP could be a participant in online shopping, auction, and electronic exchanges. The DM works with Attributes that model the notion of Value for the TP.

An Attribute describes the value that a TP gives to a particular aspect of a product. For instance, a customer buying a travel ticket online may measure the value of that product as several attributes like time-of-departure, day-of-departure, month-of-departure, time-of-arrival, day-of-arrival, month-of-arrival, number-of-stops, airline, type-of-aircraft, type-of-seat etc.

Each attribute has values defined over one or more ranges. For instance, number-of-stops may have a range of 0 to 3 with intervals of length 1. Thus, the range 0 to 3 has 3 intervals 0-1, 1-2, 2-3. Attribute values are defined at the highest discrete point in a range which in this case is 1, 2, or 3. It is possible for attributes to have more than one

range. For instance, an airline that flies Monday-Wednesday, Friday-Saturday has 2 ranges with 3 values in the first range and 2 in the second range. It can be assumed that in all intervals in any range are of fixed length. A non-fixed interval length range can be broken down into a collection of intervals with fixed length ranges.

Attributes may be dependent or independent. For instance, for each Day-Of-Week attribute value there are several Time-Of-Day attribute values. Here the Time-Of-Day attribute is dependent on the Day-Of-Week attribute. An attribute that does not depend on any other attribute is called an independent attribute. In this discussion one dependence tree structure is created by creating clever or arbitrary dependencies between attributes.

The Attribute data structure and the associated operators are shown in the attached code of the provisional application.

The system also provides a mechanism to achieve differential pricing based upon attributes and their values. For each attribute value, a Delta is assigned which basically describes what the TP would pay if the product that the TP buys or sells includes this attribute value. The Delta is simply described as a number that could translate to money or any other quantifiable entity. For instance, a traveler may be willing to pay 50$ more for an airline ticket that involves 0 stops or 50$ less for a ticket for each stop that the flight involves.

Attributes may be Implicit or Explicit. Explicit Attributes are those that are explicitly described by the TP. Implicit Attributes are those that are implicitly discovered by the system based upon data obtained from information gathering or other business intelligence applications.

It is assumed that Attributes may be managed, stored, and served by an Attribute Manager (AM) process or Server. For each attribute the AM contains information

about the name, ranges, and intervals of the attribute. Note that this AM is not dependent on any particular eCommerce or trading system that the TP is participating in.

The inventors' system creates a TP Attribute tree (TP Tree) which encapsulates the dependency between all the attributes that the TP cares about for a particular product. It is described how the TP Tree is created and managed for a product. The Tree accommodates both dependent and independent attributes for implicit and explicit types of attributes.

A collection of TP trees wherein within each tree the nodes are organized based upon dependency may be merged into a single TP tree. This may be required to enhance searching.

Figure 1 shows a trading partner (TP) attribute Tree. The root 1 of the tree specifies the product that the TP is interested in trading. The base product 16 is an airline ticket from San Francisco to New York. The base value 14 is $300. Every node 2-10 of the tree represents an attribute name and value. Any path from the root of the tree to a leaf 20 specifies a single product. This path is a path of attribute names and values. In the case of a tree with explicit attributes (where the client has explicitly specified the attributes and values) one could load the root of the tree with the "basic product value"(BPV) that the TP specifies for the product. The edges are labeled with delta-values 12 (+ or -) by which the TP modifies the product value when the subpath from the root 1 to the node at the end of the edge is chosen. For example, in figure 1, the Delta Product Value (DPV) from node 1-2 is +30, while the DPV from node 1-6 is the sum of DPVs from 1-2, 2-6 = +30 -30 = 0. The DPV at the leaf of the tree specifies the DPV along the entire path from the root to that leaf.

Figure 2 shows a path from the root to a leaf of an attribute tree. Attributes a1 to a6 are on a path to leaf 32 for tree 30. The value of the product at leaf 32 is the sum of the attribute values a1, a2, a3...a6, plus the base value.

IMPLICIT ATTRIBUTE TREES:

As mentioned above implicit attributes are those whose delta product values (DPVs) are not explicitly available in the data. Consider the example of an online airline reservation system like Travelocity. One can find prices for air tickets from City A to City B for different airlines, different arrival, departure dates and times, for different classes of services and for different number of stops between the two cities. An automatic information gathering system can repeatedly query the Travelocity server and obtain this information and store this information in a store as in a relational database. From the data stored in this store it can infer the attribute tree using complex sequel queries.

Figure 3 shows how the data from multiple data sources or applications can be obtained implicitly. A data gathering system 34 and integration system 38 populate relational store 36. Notice that any other kind of store may be used to store the information. In the example shown in figure 3, five pieces of data are stored for a particular airline in store 36 - from city, to city, time of day, day of week, fare. Using this information the Implicit Attribute Tree is constructed as follows. Heuristics may be used to build this tree efficiently. The table may look like this:

FROM	TO	TimeOfDay	DayOfWeek	Fare
SFO	NY	Morning	Monday	240$
SFO	NY	Evening	Monday	200$
ORL	SEA	Morning	Wednesday	110$

The following steps may be followed in building the Attribute tree:

Step 1: The First values in the first two columns give the products. For instance, (SFO, NY) travel could be one product and (ORL, SEA) travel could be another product.

Step 2: Then the names of the columns (Attribute Metadata, in general) are identified and checked with the AM. This helps figure out the ranges and the intervals for that attribute. For instance, name of column 3 "Time of Day" (or the data type of the data in column 3) is checked with the Attribute Manager to determine the ranges and the intervals for that attribute.

Step 3: For each attribute whose range has m intervals a bit-vector of length m is created. For example, for DayOfWeek attribute a 7-bit vector is created with one bit for each day from Monday to Sunday. The I-th attribute value is represented by turning on the I-th bit in the bit vector.

Step 4: The bit vectors are concatenated in the sense that if there are N attributes each with range of length m a bit vector of length N*m is created. This means that to represent and check all possible combinations of attribute values m**N possible bit vectors may be created. Each row in the relational table shown above creates a new bit vector.

Figure 4 is a block diagram of an attribute-tree builder. Statistical system 48 reads relational data tables 40, 42, 44, which may be located on seller web sites. Online Analytical Processing (OLAP) system 46 may also be used to mine the data in relational tables 40, 42, 44 to determine delta values and attributes that determine the price of products in a product family. Systems 48, 46 then populate attribute trees 49.

To avoid creating an exponentially large tree, heuristics may be used to create a sparse, but yet, reasonable representative of the tree. Standard statistical sampling techniques

may be used to do this. Figure 5 shows one embodiment of a heuristic to create an attribute tree.

A table of fares (or values) is accessed, step 51. A parameter K is selected in step 52. K represents number of rows to select in steps 54, 55. A base value is selected, step 53. For instance, it may be the lowest fare in the table or just an average of all the fares in the table.

In step 54, K rows are selected from data store R that have the largest values. In step 55, the K rows with the lowest values are selected. Thus 2*K rows are selected from the table (store R) where K rows represent the top K fares and the other K represent the bottom K fares. K is fixed a priori in step 52.

The parameter P is selected, step 56. From the remaining rows, another P rows are randomly chosen based upon the values in columns corresponding to the leaf level nodes, step 57.

From the remaining rows, for each subtree rooted at the node above the leaves, representative rows are sampled based upon the variance of the data at the leaf level, step 58.

In step 59, the product values (DPVs) are computed for each path (each row picked from steps 54, 55, 57. The base product value must be carefully chosen in step 53. The tree is then re-balanced.

Note that other heuristics may be applied to produce a representative sparse enough tree of the big attribute tree. When such a tree is looked up for an attribute value and/or a Delta associated with it, either it is found or it is missed. If the tree is a sparse representative of the actual tree, the missing element may be just not represented in the tree. The lookup algorithm should return a Delta that can be "guess"ed from the sibling

of the node where the looked up attribute value could have been. In order to be able to do this, one needs to optimize the implicit attribute tree.

OPTIMIZING IMPLICIT ATTRIBUTE TREES - FIG. 6

Figure 6 is a procedure for optimizing an attribute tree. Optimizing the attribute tree keeps high variance nodes on top and low variance nodes at the bottom. It is preferable for the sake of reduced representation of the attribute tree that the attributes whose values have low variance appear at the bottom of the tree and the attributes whose values show high variance be at the top of the tree.

In order to do this the attributes are broken down into two types - Category types and Measure types. Measure types are those in which the values can be placed in a monotonic order. For instance, time-of-day, ages-of-people fall in this category. Category types include types like city-of-sale or shipping-company. Category types have values that are discrete in nature and there is no obvious monotonic ordering between attribute values. The choice of these types is somewhat arbitrary because while one can think of "city-of-sale" as a category type, one can look at the geographical location of the cities and identify a monotonic ordering based upon longitude or latitude. Category types can have hierarchies like city->state->country->continent or day->month->quarter->year.

One could run column-wise queries to do the following:

1. to identify the attributes that may be supported.
2. To identify the distribution of the data over that attribute type (steps 62, 64).

Once the distribution is identified, variance of the distribution can be identified, step 66. Then attributes with low variance can be moved towards the bottom of the tree and the attributes with higher variances can be moved towards the root of the tree, step 68. This is done by comparing columns pair-wise to see which columns results in an attribute that is the parent of the attribute created by the other column. Given two

columns C1, and C2, let A1 and A2 be the corresponding attribute types to be created. The node of type A1 is a parent of the one of type A2 if the following is true: The average of the variances of the C2 values for each C1 value is less than the average variance of the C1 values for each C2 value. (low variance nodes lower).

In the case where there are too many values for a category type data with hierarchy, it may be hoisted to a higher level datatype to reduce the number of attribute values. For instance, if there are 1000 different occurrences of city-of-sale over 10 states one may choose to keep only state-of-sale information. For Measure type data one may combine close measures into measure ranges. For instance, instead of keeping age information by age a programmer may decide to keep this information by age groups. This allows building compact representations of the attribute trees.

EXPLICIT ATTRIBUTE TREES:

As mentioned before Explicit Attribute trees are created based upon explicitly provided preferences of the TP. For instance, an air traveler might have the following decision logic:

If flight on Monday - delta = +30 else if on Sunday delta = -30 else if on Tuesday delta = +50

If flight in Morning delta = +30 else if in the afternoon delta = +50 else if at night delta = -10

...

From the logic given above an explicit tree may be created. It is preferable that the tree be sparse in nature for purposes of efficient processing.

CUSTOMIZING THE TRADING PARTNER TREE

The Trading Partner Tree can be transformed into simpler representations when a TP participates in a trade. The Trade Tree is a restriction of the TP tree where the TP is participating in a trade of a particular product where some of the attributes and attribute values are not required or supported.

USAGE OF THE ATTRIBUTE VALUE SYSTEM FOR INTELLIGENT E-COMMERCE APPLICATIONS

The TP Tree, implicit or explicit, is used by a Trading Partner (end user or business) in participating in an electronic commerce application. For instance, the inventors' system allows an electronic business to attach true values to items based upon the attributes specified by the user. It allows the businesses to employ buyer-specific pricing of items based upon the attributes specified by the buyer. It also allow businesses to target-sell items based upon user preferences.

Figure 7 is a flowchart for a Business-to-Consumer (B2C) application of the attribute trees. User visits a Vendor site to buy a product P, step 70. The user provides specification for product P along with a list of attributes and values that he/she cares about, step 72.

The user specifies a baseline amount that he is willing to pay for the product, steps 74, 76. (This is an optional block).

The user may optionally specify how much he values each attributes, step 78. In one embodiment the user specifies "Delta" amounts for zero or more of the attribute and attribute values, steps 80, 82, 84.

The vendor identifies and sells the product instance that the prospective buyer values the most (the one that computes to the maximum on the sum of the "Delta" values on

all the attributes; an alternative to the "Delta" amounts may be used like some other numeric scheme negotiated between the buyer and seller). This is shown in steps 86, 88, 89.

The vendor could even dynamically place a true value on the product instance based upon the buyer attribute preferences.

AUCTION SCENARIOS

The attribute system enhances electronic auction systems where the bidder and the seller can price and/or determine the value of an auction item based upon multiple attributes of the item wherein the overall value of an item is determined by the value attached to individual attributes related to the item. In a forward auction system there is a single seller and multiple prospective buyers.

Figure 8 is a flowchart of an auction system enhanced by the attribute-structuring facility. Given a list of attributes associated with a product being auctioned, the method of figure 8 helps a prospective buyer to choose a bid based upon the attributes that he/she values the most. The seller puts up an item for auction with its attributes and attribute values, step 90. Each bidder has a baseline value for the item, and the maximum bid prices is set to the baseline value, steps 92, 94. For each attribute that the bidder cares about, step 96, the bidder has a delta to be added/subtracted if the product includes/excludes that attribute, step 98. By going through all the attributes of the product (steps 95, 96) and adding/subtracting the deltas from the baseline value, the system computes the maximum bid price for the bidder, step 99. The bidder bids on the product between the baseline and the maximum bid price for what he/she thinks is the real value of the product.

Figure 9 is a flowchart of an optimal product/seller selector that helps a bidder. In an auction system, where multiple products are being auctioned and each of the products have listed associated attributes, it enables the prospective buyer to bid on items that

satisfy most of his/her attributes. Here the bidder can pick between multiple sellers who are selling "similar" products but different support for attributes that the bidder cares about. The bidder looks for a product whose attributes have the most overlap with the attributes the bidder cares about and bids on that/those product(s).

A bidder Bi looking for a product P finds several sellers auctioning the product in various configurations, steps 102, 104. For each seller, the seller's attribute tree is compared to the buyer's attribute tree, steps 108, 109. The attribute names, values, and ranges may not exactly match, so they may need to be normalized. The degree of overlap detected in step 110 is stored for that seller. The attribute tree comparison is repeated for the other sellers with the same buyer's attribute tree, steps 112, 106-110. The maximum overlap is found, step 114. The maximum overlap indicates an optimal product configuration by one of the sellers that most closely matches the desired configurations of the buyer. The buyer can then bid or purchase the product with the optimal configuration from the most-closely overlapping seller, step 116.

A selling party may provide augmented products where the product being auctioned is dynamically augmented with new attributes to satisfy the attributes that are valued the most (or by most prospective buyers). Here the seller goes through all the attributes that bidders for a particular product care about and augments the product with those attributes and values that make it the most marketable item.

Figure 10 illustrates dynamic augmentation of a product with the attributes most desired by the potential buyers. A matrix of attributes and possible attribute values is created for a product, step 130. Preference data is collected from several prospective buyers or bidders, step 132. For each potential buyer Bi, each attribute is checked to see if the user has entered a preference or attribute value for that attribute, step 138. When the buyer has entered an attribute value Vjt for attribute Aj, a counter for that attribute value is incremented, step 140.

An inner loop increments the loop variable j for all attributes, steps 142, 144. An outer loop increments the loop variable i, for all potential buyers, steps 146, 148. The loop variables are initialized in steps 134, 136. Thus all variables are checked for all potential buyers. The attribute counters incremented in step 140 are compared to find the maximum count(s), and the corresponding attributes are used to augment the product, step 150. The maximum counts corresponds to the most-desired attributes for the product.

The above scenarios are only representative uses of the inventors' system in some auction systems. Many other scenarios and uses are contemplated.

AGGREGATION

Aggregation is used in auctions to aggregate small buyers and sellers to match them up with a seller or a buyer who can only sell to or buy from large parties. Attributes can enhance the aggregation process by allowing aggregation of buyers and sellers not just based on products but based upon similarity in the attributes they value the most.

Java interface classes and pseudo code were attached to the provisional application.

ADVANTAGES OF THE INVENTION

The inventors' multi-attribute model is well-suited for electronic exchanges. The attribute trees provide a way to express the many options or configurations of products in a product family. The attribute tree is a convenient mechanism to receive, store, and compare multiple product configurations from many buyers and sellers. The attribute tree is a simple yet powerful structure for storing many attributes of a product family.

The invention provides a system that allows a trading partner to model a product or augment a product with multiple attributes statically or dynamically based upon the other participants in the trade. The system discovers attributes from implicit set of data,

optimizes attribute representation when large number of attributes are present, and also allows dependencies between attribute values. The system can be used in a Business-to-consumer or Business-to-Business scenario.

ALTERNATE EMBODIMENTS

Several other embodiments are contemplated by the inventors. The increment of the counter can be by more than 1, or a negative increment (decrement) can be substituted. Non-binary counting schemes such as Gray Code can be substituted. Rather than incrementing the counter, the value that the buyer places on an attribute can be summed. Thus buyers who place more value on an attribute than other buyers can have a larger weighting for that attribute when the optimal set of attributes is determined.

Attributes may also be discovered from the preferences of the trading partners. For example, a traveler may specify that he "prefers" flying in the morning and "detests" flying at night. These may be used to express time-of-day attributes with the value of "morning" with a positive delta of 20% of the base value of a ticket and the value of "night" with a negative delta of 50% of the base value. The attribute value of "afternoon" may have a delta of 0.

The foregoing description of the embodiments of the invention has been presented for the purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise form disclosed. Many modifications and variations are possible in light of the above teaching. It is intended that the scope of the invention be limited not by this detailed description, but rather by the claims appended hereto.